alto <= 25000 * conv_integer(sel);
pwmsal <= '1' when conta <= (50000+alto) else

```
-- Este código sirve para controlar tres posiciones  de un servo
-- con anchos de pulso de 1ms, 1.5ms y 2ms (0°-90°-180° en un genérico)
-- con periodo fijo de T = 10ms (f=100Hz)
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-----Entidad-----
entity pwm is
    Generic( N : integer:=24); --N+1 es el máximo de bits para clkdiv
   Port ( clk : in  STD_LOGIC; --señal de reloj de 50MHz
          led : out  STD_LOGIC_VECTOR (2 downto 0); --leds testigos de la posición
          pwmsal : out  STD_LOGIC); --salida de pwm hacia el servomotor
end pwm;

-----Arquitectura-----
architecture pwm of pwm is
--señal para el contador de uno a 500,000
          signal conta: integer range 1 to 500E3:=1;
--señal para el divisor del reloj
          signal clkdiv: std_logic_vector(N downto 0);
--señal para el selector que controla las posiciones del servo
          signal sel: std_logic_vector(1 downto 0);
--señal que guarda el valor que se suma para el tiempo en alto
          signal alto: integer;

begin

--divisor
div: process(clk)
begin
          if clk'event and clk='1' then clkdiv <=clkdiv + 1; conta <= conta + 1;
                    if conta > 500E3 then conta <= 1;
                    end if;
          end if;
end process div;

--contador para el selector
```

```
conteo:process(clkdiv(N))
begin
        if clkdiv(N)'event and clkdiv(N)='1' then
                if sel = "10" then sel <= "00";
                else sel <= sel+1;
                end if;
                        case sel is
                         when "00"   => led <= "001";
                         when "01"   => led <= "010";
                         when "10"   => led <= "100";
                         when others => led <= "111";
                        end case;
        end if;
        --se calcula el valor de la señal alto
        alto <= 25000 * conv_integer(sel);
end process conteo;

--generación de pwm en base al valor de alto(que depende de sel)
pwmsal <= '1' when conta <= (50000+alto) else
        '0';

--fin del programa
end pwm;--fin
```

El código que se escribió para realizar la implementación en la tarjeta, conocido como "archivo de restricciones de usuario" (User Constrains File) guardado con extensión *.ucf es el siguiente.

```
##############################################################
#asignación para la nexys 2 del servo automático 3 posiciones
#reloj
net "clk" loc = "b8" ; #reloj de 50MHz
#leds
net "led(0)" loc = "j14" ; #al led0
net "led(1)" loc = "j15" ; #al led1
net "led(2)" loc = "k15" ; #al led1
#salida al servo
net "pwmsal" loc = "M15" ; # al JA4, o bien "t17" ;#a JB4, o bien "m13" ;#al JB1
##############################################################
```